AFOSR-TR-95

# REPORT DOCUMENTATION PAGE

0672



urces, of this erson

| 1. AGENCY USE ONLY (Leave blank) | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | | FINAL/30 SEP 93 TO 31 MAR 95 |

**4. TITLE AND SUBTITLE**
DISTRIBUTED OPTIMIZATION IN AIRCRAFT MISSION SCHEDULING

**5. FUNDING NUMBERS**
3484/BS
F49620-93-1-0528

**6. AUTHOR(S)**
KENDALL E. NYGARD

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
NORTH DAKOTA STATE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE &
OPERATIONS RESEARCH
FARGO, ND 58105

**8. PERFORMING ORGANIZATION REPORT NUMBER**

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
AFOSR/NM
110 DUNCAN AVE, SUTE B115
BOLLING AFB DC 20332-0001



**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
F49620-93-1-0528

**11. SUPPLEMENTARY NOTES**

**12a. DISTRIBUTION/AVAILABILITY STATEMENT**
APPROVED FOR PUBLIC RELEASE: DISTRIBUTION IS UNLIMITED

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** *(Maximum 200 words)*

Decision support tools for customized routing and scheduling of aircrct to support mission-critical travel has been developed and implemented. The system has been installed for operational use by (USAFE).

19951018 058

DTIC QUALITY INSPECTED 5

| 14. SUBJECT TERMS | | | 15. NUMBER OF PAGES |
|---|---|---|---|
| | | | 16. PRICE CODE |
| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
| UNCLASSIFIED | UNCLASSIFIED | UNCLASSIFIED | SAR(SAME AS REPORT) |

# Distributed Optimization in Aircraft Mission Scheduling

## Final Report

May 25, 1995

Grant F49620-93-0528
North Dakota State University

Kendall E. Nygard
Department of Computer Science and Operations Research
North Dakota State University
Fargo, ND 58105

# 1. INTRODUCTION

The investigation concerns the development and implementation of distributed optimization models and related decision support tools for customized routing and scheduling of aircraft to support mission-critical travel. The resultant software, called the DAKOTA system, includes an extensive database management system, an advanced graphical user interface (GUI), as well as the mathematical optimization scheduling procedures. DAKOTA has been installed for operational use by the United States Air Forces in Europe (USAFE). The primary accomplishments of the project are the development of and solution procedures for the optimization model, and the architecture and implementation of the procedures for schedulers at distributed locations and interconnected by a computer network. The graphical user interface provides easy and intuitive access to the various tools of the system.

# 2. THE AIRCRAFT SCHEDULING APPLICATION

The procedures developed apply to general settings in which there is a set of scarce resources that are capable of carrying out tasks. In the passenger airlift application, the resources are aircraft, and the tasks are passenger requests for travel support.

USAFE manages a fleet of aircraft that supports passenger travel in Europe, all of Africa, and much of the former Soviet Union. Passengers include officers and government officials carrying out missions of critical importance to the military.

A request for travel is specified by its point of origin, desired departure time, destination, desired arrival time, number of passengers, and mission priority. The available fleet is composed of differing aircraft types (in the USAFE problem alone, there are C-12, C-20, C-21, T-43, UH-1N and C-130 aircraft). The aircraft vary widely in several aspects, including capacity, endurance and speed. Multiple schedulers create missions that support the travel. This involves assigning contingents to aircraft, preparing flight itineraries, coordinating activities with personnel responsible for crew scheduling, and communicating with passengers being scheduled. Prominent problem characteristics are given below.

1) Several schedulers work semi-autonomously, but are allocating commonly held airlift resources.

2) Aircraft can simultaneously service multiple travel requests.

3) Travel requests have varying priorities.

4) Both immediate and advance and immediate travel requests occur.

5) Some travel requests can pre-empt others, requiring that rollback be supported.

6) Aircraft are constrained in capacity.

7) Aircraft types have differing characteristics, including airspeed, endurance, and cost.

8) There are multiple aircraft of each type.

The decision support environment is inherently distributed, because multiple schedulers work in parallel, each responsible for scheduling travel requests on dedicated aircraft. From an optimization view there are several objectives: maximize the number of travel requests supported by priority class, minimize mileage traveled, and minimize inconvenience to the passengers (extra stops and waiting for example). If the schedulers work independently, they may each produce locally optimal results within their limited domains, but the coalesced work is likely to be far from optimal.

The overall steps in mission scheduling are as follows;

Step 1. Select Travel Requests and Missions for Scheduling

Step 2. Select a Request Ordering Criterion and an Insertion Evaluation Criterion

Step 3. Order the Selected Requests Firstly by Priority Class, and, Secondly, by the Request Ordering Criterion

Step 4. Loop through the Requests in Order. For each request and for each Mission, use Constraint Propagation to calculate and Generate the Schedule Associated with Inserting the Request in the best Feasible Location as Measured by the Insertion Location Evaluation Criterion

Step 5 Formulate a Set Partitioning Problem from the Generated Schedules. Apply the Pricing Heuristic to Select a Subset of schedules and Present them to the Scheduler.

## 3. CONSTRAINT PROPAGATION AND SCHEDULE OPTIMIZATION

A Client-server distributed computing model was adopted to provide the means of coordinating the work of the schedulers. Within this paradigm, the computer used by each scheduler is handled as a "client". The "server" handles access to the database, and provides mechanisms for coordinating the activities of the schedulers, with the goal of producing schedules that are close global optimality. The clients benefit, in their local decision making, from the server's global view of aircraft allocation and contention for available aircraft. The server functions range from as fundamental as ensuring that multiple schedulers do not schedule the same aircraft for conflicting missions in the same time period, to coordinating a distributed "bidding" process for aircraft time under contention. Each scheduler is supported by a sophisticated mathematical optimization model that employs constraint propagation to generate the ramifications of time-windows on travel requests and aircraft availability, and a set partitioning solver [Nygard, 1993; Sycara et al, 1991; Sadeh, 1993]. These decentralized solutions are evaluated for resource contention by the server.. Steps 1-4 provide a collection of candidate solutions that can be formulated in a set partitioning problem. The effectiveness of the procedure is, in part, due to the use of request ordering in Step 3, a technique that has origins in the direct generation of schedules in job-shop scheduling problems.

The extended set partitioning problem (ESPP) is stated mathematically as follows:

***Indices.***

i = 1, 2, ....,n aircraft
j $\varepsilon$ J(i), the set of feasible schedules for aircraft i
k = 1, 2, ....,l requests for travel

***Data***

$c_j$ = cost of schedule j

$v_{kj}$ = 1 if schedule j supports request k, 0 if not

$p_i$ = penalty cost for an idle aircraft

$q_i$ = penalty cost for overscheduling an aircraft

$r_k$ = penalty cost for not accommodating a travel request

$s_k$ = penalty cost for overaccommodating a travel request

***Decision Variables***

$y_j$ = 1, if schedule j is selected, 0 if not

$u_i$ = 1, if aircraft i is not utilized, 0 if utilized

$\sigma_i$ >= 0 and integer, indicating overscheduling of aircraft i

$w_k$ = 1, if reqeust k is not supported, 0 if supported

$\alpha_k \geq 0$ and integer, indicating multiple support for request k

***Mathematical Formulation***

$$\text{Min} \sum_j c_j y_j \sum_i (p_i u_i + q_i \sigma_i) \sum_k (r_k w_k + s_k \alpha_k)$$

Subject to:

$$\sum_{j \varepsilon J(i)} y_j + u_i - \sigma_i = 1 \qquad \text{for each aircraft i}$$

$$\sum_j v_{kj} y_j + w_k - \alpha_k = 1 \qquad \text{for each travel request k}$$

$y_j, u_i, w_k = 0$ or 1

$\sigma_i, \alpha_k \geq 0$ and integer

In a set partitioning model, the goal is to identify a collection of variables (columns, representing aircraft missions) that partition the rows (representing travel requests) so that the objective function value is minimized. Since the number of possible columns that could be generated is exponential in the number of travel requests, it is important to develop criteria that restrict the number of columns generated, yet retain columns that represent good schedules. A primary accomplishment of the project was the development of mathematical methodologies for avoiding the generation of excessive numbers of columns. The article attached in the Appendix provides a description of how the hierarchical constraints at several levels propagate to neighboring levels and within the levels to dramatically limit the number of generated columns, yet retain high quality solutions.

The set partitioning solver utilizes a marginal pricing procedure inspired by linear programming. The procedure begins with an initial feasible solution in which each available aircraft is empty and flies no mission. For each aircraft in turn, available passenger loads (columns) are priced for marginal potential to improve the solution, and iteratively assigned to the aircraft. When assignments of passenger loads are made, the aircraft receiving the load may, in effect, be usurping passengers previously assigned to another aircraft. The process terminates when no further improvement is possible. The solution procedure is shown below, in set notation which surpresses the details of data structures and the mechanics of solution updates. The abstract data types could support alternative implementations.

```
Global: allLegs, allPlanes, allRoutes

findBestRoute(plane, routesPicked, route, incProfit, totProfit ) {
/* plane must not fly an already picked route */

   route=none; bestRP=routesPicked;

   for each r in allRoutes: plane flies r {
     curP=profit(r);
     if(curP< incProfit) break;

     rp2=routesPicked;

     for each s in routesPicked {
       s2 =: legs(s2)==legs(s)-legs(r);
       rp2=rp2-s+s2;
       curP+=profit(s2)-profit(s)
     } /* s */

     if(curP> incProfit) {
       incProfit=curP;
       bestR=r;
       bestRP=rp2;
     }
   } /* r */
} /* findBestRoute*/
```

```
findRouteSet(routesPicked, totProfit) {
/* find set of routes from scratch */

  totProfit=0;
  routesPicked= empty;

  for each plane in planeSet {
  /* order to be determined */
    incProfit= -1;
    findBestRoute(plane, routesPicked, route, incProfit, totProfit);
  } /* p */
} /* findRouteSet */


improveRouteSet(routesPicked, totProfit) {
/* tries to improve existing solution */

  do {
    converged=1;
    for each plane in planeSet {
    /* order to be determined */
      /* save current data, remove route of plane */
      oldRoute=routeOf(plane);
      incProfit=profit(oldRoute);
      routesPicked-=oldRoute;
      oldTotProfit=totProfit;
      totProfit-=incProfit;

      /* try to find a better route */
      findBestRoute(plane, routesPicked, newRoute, incProfit, totProfit);

      if(newRoute==none) routesPicked+=oldRoute, totProfit=oldTotProfit;
      else converged=0;
    } /* plane */
  } while(!converged && !done);
} /* improveRouteSet */
```

## 4. TESTING AND CONCLUSIONS

The procedure was extensively tested on randomly generated problems for verification purposes, and on actual Air Force data obtained from the USAFE operation. A testing difficulty with actual data is that good records are not available for travel requests that are not supported--only for travel that was supported and the associated missions actually flown. With this difficulty understood, it is of interest that the testing with actual data reveals that in most historical situations the passengers whose travel was supported could have been accomplished with significantly fewer aircraft missions than were flown in practice. Had data for unsupported travel requests also been available, it is possible and likely that ways for the fleet to have supported the travel of additional passengers would have been identified by the model. The primary goals of the project were met: an effective and efficient column generation procedure was developed, and a fast solver and effective solver was implemented. Additional challenge lies in identifying ways in which the efforts of individual schedulers can be more effectively coordinated, to produce schedules even closer to global optimality.

## 5. REFERENCES


Appelgren, L. H. A Column Generation Algorithm for a Ship Scheduling Problem. Transportation Sci., 3:53-68, February, 1969.

Appelgren, L. H. Integer Programming Methods for a Vessel Scheduling Problem. Transportation Sci., 5:64-78, February, 1971.

Bell, Colin E. and Austin Tate. Using Temporal Constraints to Restrict Search in a Planner. Technical Report, Artificial Intelligence Applications Institute, University of Edinburgh, December. 1984.

Brown, Gerald G., Glenn W. Graves and David Ronen. Scheduling Ocean Transportation of Crude Oil. Management Science, Vol. 33, No. 3, March, 1987.

Darby-Dowman, K. and G. Mitra, An extension of Set Partitioning with Applications to Scheduling Problems. European Journal of Operations Research 21, 1985

Fox, Mark S., Norman Sadeh and Can Buykan, "Constrained Heuristic Search", Technical Report, Robotics Institute, Carnegie-Mellon University, 1989

Fox, Mark S. and Stephen F. Smith. ISIS: a Knowledge-Based System for Factory Scheduling. Expert Systems 1(1):25-49, July, 1984.

Garfinkel, R. S. and G. L. Nemhauser. The Set Partitioning Problem: Set Covering with Equality Constraints. Oper. Res., 17:848-856, 1969.

Le Pape, Claude and Stephen F. Smith. Management of Temporal Constraints for Factory Scheduling. Temporal Aspects in Information Systems. C. Rolland, R. Bodart, and M. Leonard (Editors). Elsevier Science Publishers B.V. (North-Holland). IFIP, 1988.

Marsten, R. E. An Algorithm for Large Set Partitioning Problems. Management Sci., 20:774-787, 1974.

Miller, David, James Firby and Thomas Dean. Deadlines, Travel Time, and Robot Problem Solving. In Proceedings of the Ninth International Joint Conference on Artificial Intelligence. Los Angeles, California, August, 1985.

Rit, Jean-Francois. Propagating Temporal Constraints for Scheduling. In Proceedings of the 5th National Conference on Artificial Intelligence. Philadelphia, Pennsylvania, August, 1986.

Ronen, D. Cargo Ships Routing and Scheduling: Survey of Models and Problems. European J. Oper. Res., 12:119-126, 1983.

Smith, Steven F. and Peng Si Ow, "The use of Multiple Problem Decompositions in Time Constrained Planning Tasks", in Proceedings of the Ninth International conference on Artificial Intelligence, 1013-1015, 1985

Smith, Stephen F., Peng Si Ow, Claude Le Pape, Bruce McLaren and Nicola Muscettola. Integrating Multiple Scheduling Perspectives to Generate Detailed Production Plans. In Proceedings of the SME Conference on AI in Manufacturing. Long Beach, California, September, 1986.

# APPENDIX

## ARTICLE WHICH PRESENTS THE TECHNIQUES USED IN CONSTRAINT PROPAGATION FOR GENERATING COLUMNS FOR THE SET PARTITIONING MODEL AND FOR SUPPORTING MULTIPLE SCHEDULERS

# Cooperative Airlift Resource Allocation and Scheduling

*Dr. Kendall E. Nygard, Ph.D.*
Department of Computer Science and Operations Research
North Dakota State University
Fargo, North Dakota 58105
Tel: 1-701-231-8203
E-Mail: nygard@plains.nodak.edu

*Richard Walker, MS*
Department of Computer Science and Information Systems
Moorhead State University
Moorhead, Minnesota 56563
Tel: 1-218-236-2384
E-Mail: walker@mhd1.moorhead.msus.edu

**ABSTRACT**
Resource allocation and scheduling problems involve tasks which must be allocated to facilities (resources), and scheduled for processing on those facilities. The resources and the tasks to schedule may be subject to constraints that restrict allocation possibilities and the time frames during which tasks can be scheduled. This work concerns resource allocation and scheduling problems in which multiple schedulers work in parallel and schedule resources held in common. The level and structure of the coordination of the work of the schedulers is a central issue. Models for maintenance, propagation and utilization of global task and resource information are presented and discussed with respect to the airlift resource allocation and scheduling problem.

**KEYWORDS**
cooperative scheduling, resource allocation, distributed computing

**INTRODUCTION**
Resource allocation and scheduling problems involve tasks which must be allocated to facilities (resources), and scheduled for processing on those facilities. The resources and the tasks to schedule may be subject to constraints that restrict allocation possibilities and the time frames during which tasks can be scheduled. This work concerns resource allocation and scheduling problems in which multiple schedulers work in parallel and schedule resources held in common. The level and structure of the coordination of the work of the schedulers is central issue. At one extreme, the schedulers work in complete independence from each other. In this situation, the scheduling agents will produce schedules that are at best optimal only within their domain of local responsibility, and the aggregate solution may be far from global optimality. The other extreme is full communication, coordination and cooperation among all scheduler activities, the equivalent of centralized single agent scheduling. This situation permits the attainment of a globally optimal scheduling solution, but may be impractical due to the workload level being inherently distributed or too large to centralize.

Although much research and development has been done in scheduling and routing, there is little that concerns developing schedules in a distributed multi-agent environment [7]. In the case of centralized or single agent scheduling, a host of issues involving contention for resources never appear. The fundamental complications in the multiple agent case are: i) currency of information retrieved for local schedule optimization and; ii) balance of resource allocation between local and global concerns. The challenge of distributing scheduling tasks among multiple agents who share resources lies in the limited view of the current resource needs and the intentions of peer schedulers. If an agent does not have a global view of the system, a good local resource allocation decision may have a negative global impact. It may even be the case that partially completed schedules may make a globally feasible resource scheduling assignment impossible. When this occurs, requests must be unscheduled, a process called schedule rollback. Excessive rollback creates scheduler inefficiency. To address the myopic view of scheduling in the distributed case, we present a model which provides, to the individual schedulers, information about availability and system wide contention for resources. Using this global information, a human scheduler or scheduling algorithm can make local scheduling decisions which are also good globally.

In scheduling executive airlift, several schedulers are responsible for arranging travel for executives who submit requests composed of one or more origin-destination pairs, called request legs. Each scheduler is responsible for a subset of travel requests and develops aircraft routes and schedules for them subject to constraints on the aircraft fleet and on the requests themselves. Some constraints are operational in nature and are due to physical and temporal restrictions on the aircraft and crews. Travel speed, aircraft capacity, endurance, and crew duty hours are examples of these. Constraints imposed by the requests include contingent size, "hard" time windows on departure and / or arrival times, and "soft" constraints concerning inconvenience to the travelers. Hard time windows are inviolate. Soft time windows can be violated at a penalty cost. Other constraints are imposed by the schedules developed by peers. The schedulers, working semi-autonomously, attempt to obtain the best possible schedules for the requests for which they are responsible, without violating existing constraints. Globally, the problem is to share the aircraft, possibly simultaneously, among requests for travel in a manner which optimally supports the requests.

We present an approach to single agent scheduling which characterizes the allocation and scheduling of aircraft among requests for travel as a set partitioning problem by constructing feasible assignments of travel requests to aircraft and representing them as set partitioning columns. Each column represents the feasible assignment of an ordered subset of the set of unscheduled request legs to a specific aircraft mission. Algorithmic tools for constraint propagation [2,5], variable ordering and value ordering [6] are employed to assist in manual schedule construction or to provide automatically generated solutions at the local scheduler level. Next, we focus on distributed computing models for including global information in local decision making. In this context, where the resources of interest are aircraft missions and the activities are requests for travel, we discuss three distributed computing models for resource allocation and scheduling. "Autonomous peer process," "Cooperative peer process," and "Client-server" models are presented in increasing order of extensiveness of the global information and coordination available to the local schedulers. For each model, mechanisms for maintaining global resource information and distributing it to the appropriate agents is described. Finally, we describe a paradigm for the distributed scheduling of aircraft missions to service travel requests which uses global information in making local scheduling assignments.

This work makes significant contributions in two areas. A constraint satisfaction based heuristic insertion algorithm is developed for single agent airlift resource allocation and scheduling. This algorithm adapts concepts from Dial-a-Ride (DARP), Job Shop Scheduling (JSSP) and Critical Path Methodology (CPM) to the problem. Methods for constraint propagation and constraint satisfaction, which are key elements in our strategy, are combined with request insertion to incrementally develop feasible schedule assignment representations. These representations may be evaluated by a set partitioning heuristic to produce schedules which meet operational objectives while satisfying constraints on both the aircraft and on the requests. The algorithm is extended to a multiple scheduling agent paradigm supported by a distributed model which facilitates system wide propagation of global information and provides a mechanism for effectively communicating resource requirements and contention. Through this model, we contribute to an improved understanding of the means of maintaining, communicating and utilizing global resource information to optimize schedule construction in a distributed setting. This decision support environment is representative of varied problem solving applications that can benefit from cooperative allocation and scheduling of resources, such as job shop scheduling and other manufacturing problems.

Software development and testing for this work is being done on DEC 5000 and Sun SPARC 10 Unix workstation computers. The software is portable to any Unix system adhering to the proposed Open Systems Foundation (OSF) standard. Code is written in ANSI C and C++. Database support for the maintenance of requests, scheduled missions, and supporting information is provided by DESc [3], a locally developed database management system for scheduling applications. The inter-process communication uses flow-controlled, connection-oriented Unix network sockets. The network of Unix color graphics workstations employing the TCP/IP network protocol suite. Work on this project is being done at North Dakota State University under Air Force Office of Scientific Research (AFOSR) sponsorship.

## DEFINING ARASP

The definition of the Airlift Resource Allocation and Scheduling Problem (ARASP) is motivated by a need to identify the salient features of the executive airlift scheduling process. Number of scheduling agents, level of agent cooperation and coordination, request (task) characteristics and constraints, and resource characteristics and constraints are feature categories which characterize the structure of the problem.

### Scheduling Agents

For small enterprises it may be reasonable for a single agent to schedule all travel requests. However, larger enterprises require that multiple scheduling agents share the task; each scheduler assuming responsibility for scheduling a subset of the requests for travel. Agents

may be located in close proximity to each other, so that personal interaction can guide the sharing of resources. On the other hand, agents and resources may be geographically separated within an installation or even throughout the world. In order to be truly flexible in our problem solution, we assume that multiple schedulers, linked by a computer network, work in parallel to service distinct requests by allocating commonly held aircraft resources.

**Support for Cooperative Scheduling**

The efficiency of individual schedulers and the global effectiveness of the schedules they create are primary concerns in the design of computer support for the allocation and scheduling process. The individual schedulers and the scheduling group, as a whole, benefit from the sharing and coordination of information regarding system wide resource availability and contention. Two identifiable elements of computer support for the system are:

* A computer network model should be implemented to encourage sharing of global information regarding resource availability, coordinate global resource allocation and support cooperative resource use.
* Algorithmic support at the level of the individual scheduler should provide optimization and decision support tools for schedule construction and for evaluation of performance measures of resulting schedules.

**Requests Characteristics**

Travel requests may be composed of multiple request legs (departure / arrival pairs), but generally represent a single contingent. Each request leg has associated time window information, described in terms of earliest/latest pickup and earliest/latest delivery. The bounds on these windows may be designated as soft or hard, depending on if they are negotiable or inviolate, resp.

The contingent size may very as individual passengers join or leave the group between stops on the itinerary, but when physically possible, the entire contingent will travel on the same aircraft. When contingent size precludes service by one craft, it is assumed that multiple requests will be submitted. For passenger convenience and security, it is considered infeasible to require a contingent to change aircraft during a request leg. Individual request legs, however, may be supported by different aircraft. Multiple requests (contingents) may be simultaneously supported by the same aircraft. Thus, stops during a request leg are permitted.

Requests are both immediate and advance notice. They may have varying priorities, which can influence the order in which they are scheduled. Since operational goals may allow pre-emption and since cancellations may occur, a means for unscheduling previously supported requests and rollback of schedules must be provided.

**Resource Characteristics**

The fleet of supporting aircraft may be non-homogeneous, composed of aircraft with varied capacity, air speed, and endurance. For the specific application, we consider small aircraft in the six to fifteen seat category. These operational constraints and others regarding flight and service time must be enforced, though there should be means to override them to provide "what-if" capability the length of time an aircraft may remain in service, required maintenance schedule, and flight, duty and rest time for crews suggest constraints on aircraft that can best be reflected by specifying our resource to be an aircraft mission. The aircraft mission becomes a loosely defined mechanism for ensuring that an aircraft returns to its home base at required intervals and allows for controlling the amount of time it is in service, in maintenance, specially allocated or unavailable for general scheduling. By focusing on the aircraft mission as our resource, we also provide improved granularity for viewing and manipulating the fleet schedule.

The above characteristics and constraints describe the paradigm for which we propose heuristic algorithms that may be used as decision support tools or as tools for automated scheduling. We then propose a model for cooperative resource allocation and scheduling in which global resource information is maintained and distributed to individual scheduling agents. In addition to providing resource information, the model provides a means for enabling negotiation for resources and for insuring the integrity of global resource schedules.

## AN ENVIRONMENT FOR COOPERATIVE SCHEDULING

Globally, the goal of our system is to efficiently allocate and schedule resources to optimally support a set of travel requests in a manner which is consistent with the constraints defined by the enterprise and by the customers. To achieve this goal, our system maintains a representation of the evolving constraints on available resources and provides this information to agents in a distributed scheduling environment. The two key elements in the system are algorithmic support for the individual scheduler and coordination of a scheduling effort distributed among multiple schedulers. Fundamental to our approach to both of these concerns is the characterization of resource availability in terms of the constraints placed on those resources by the enterprise, by the tasks for which support has already been scheduled and by resource requirements of individual schedulers for meeting the demands of the tasks they must schedule. We form a constraint hierarchy whose root represents constraints on the entire

aircraft fleet and whose leaves are constraints on individual travel request legs.

At the global level, we maintain information, in the form of fleet, aircraft and mission constraints. These constraints represent the flexibility of resources to accommodate additional requests for travel or temporal schedule shifts. They are a function of operational considerations and of constraints on request legs already being supported by the given mission. Temporal constraints can be intuitively represented as time windows on the activities of the resource for supporting the tasks assigned to it. This global information is provided, system wide, to schedulers as they consider resources to meet the needs of their unscheduled travel requests.

Individual schedulers select resources, guided by aircraft characteristics, customer needs and preferences, resource availability and schedule flexibility. The schedulers are required to request a resource and an associated feasible time window before making schedule changes to it. This provides a mechanism for determining the availability of a resource and for informing peer schedulers of resource use intentions. It also imposes additional constraints in the constraint hierarchy which will protect the reserved mission and associated reserved time window while the scheduler is making changes to it at the local level. By imposing these additional constraints, the local scheduler is essentially asking for a lock on a resource over a commonly agreed upon period of time. Only after the reservation is granted, can the local scheduler be certain that changes to the local schedule of the resource will not affect nor be affected by changes to the rest of the hierarchy. It is this step that bridges the gap between the individual, autonomous scheduler and the group of cooperative schedulers who share resources. Figure 1 illustrates the propagation of constraints throughout the constraint hierarchy, both within and between levels.

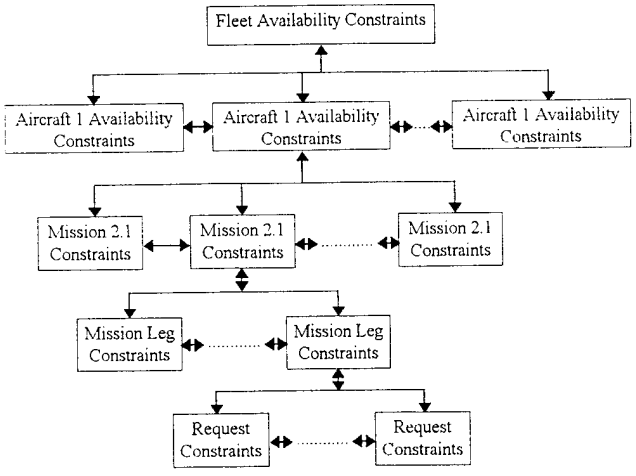


Figure 1: Propagation of time window constraints in ARASP. Effects of constraints are propagated up and down the hierarchy, as well as laterally through a level.

Resources made available at the individual scheduler level and the requests the agent must schedule onto them are also characterized by their constraints. The resources reserved by the scheduler cannot be extended beyond the constraints negotiated at the time they are acquired. Feasible assignments of tasks to resources involves producing a schedule which simultaneously satisfies the constraints on both the task and the resource. Thus, the search for feasible, and ultimately optimal, local schedule assignments is directed by those constraints imposed on the resources and on the tasks.

The degree to which the distributed model imposes restrictions on the acquisition of resources and their associated time windows determines the involvement of global decision making in the local scheduling process. Potential exists to provide a spectrum of control from the simple locking approach described above to an autocratic system that dictates which resources should be utilized to service subsets of travel requests. Our approach relies on the cooperative nature of the schedulers to negotiate reasonable time windows for their reserved resources.

The propagation of constraints in the constraint hierarchy is pursued further in the discussion of support for the individual scheduler. There we give an overview of the algorithmic support for local determination of resource schedules.

## SUPPORT FOR THE INDIVIDUAL SCHEDULER

Whether there is a single scheduling agent or multiple cooperative schedulers, we view the actual assignment of a task to a resource as being done by an individual agent. To simplify our discussion, we concentrate on the temporal restrictions due to mission time windows and request leg time windows. The foundation of our scheduling process in either mode, therefore is a mechanism for a single agent to schedule a selected set of request legs onto a set of time constrained aircraft missions.

Schedules for resources may be incrementally constructed over time, with actual scheduling done to meet specific lead time requirements of some tasks. The philosophy that guides our approach to scheduling of request legs in the executive airlift problem is that previously scheduled requests should be minimally impacted by the addition of new, unscheduled, requests to a common mission. With this as a primary constraint on creating request / mission associations, our heuristic uses an insertion approach to create assignments of request legs to missions which do not violate time window conditions on existing request leg originations and terminations. Barring violation of capacity constraints, a request leg may be serviced by an existing mission leg, or it may be necessary to create additional stops in the mission schedule to accommodate the request. Time window bounds for arrivals and

departures of individual request legs may be categorized as "soft" or "hard." A hard time window bound is one whose violation will cause an infeasibility in the constructed mission. Insertion of mission legs to service request legs which would violate existing time window constraints are deemed infeasible and are not considered further by the algorithm. Violation of soft time window produces inconvenience to the customers, but does not cause infeasibility. A request leg assignment causing a mission to violate a soft window bound for one or more of its request legs will incur a penalty for doing so, which will reduce the offending assignment's value.

Given a set of resources (aircraft missions and corresponding operational constraints) and a set of tasks (unscheduled flight requests composed of one or more request legs), the single agent scheduling algorithm performs the following activities:

- Orders request legs for insertion, based on schedule performance considerations,
- Generates Set Partitioning Problem (SPP) columns representing feasible ordered insertions of requests into aircraft schedules and their associated values, and
- Solves the Set Partitioning Problem to produce updated aircraft schedules.

The ordering of the travel request legs may be viewed as preprocessing, while the Set Partitioning Problem solution performs a postprocess function. The heart of our algorithm focuses on the propagation and satisfaction of constraints to generate representations of feasible task-to-resource assignments. Here, we focus on the ordering and column generation components of the process.

**Variable Ordering**

Travel requests can be scheduled as they are received or may be short-term or long-term batched. Since insertion algorithms are inherently myopic, the order in which insertions are performed is critical to the quality of the resulting schedules. Batching requests allows the application of variable ordering [6] techniques to prescribe a request leg order for insertion consideration. The particular ordering employed depends on the operational and strategic goals of the scheduling effort. "Best Fit" ordering is an attempt to increase utilization of resources on flight legs which are already scheduled. Those requests which may be serviced by existing flight legs are considered for insertion early, with a preference given to those which will most nearly fill an aircraft for the request leg path. "Earliest First" ordering can be used to delay scheduling of those requests farther out on the scheduling horizon and thereby maintain flexibility of schedules until more requests for that time frame are in hand. These schemes may be combined or used separately to address those measures of performance deemed most critical to the enterprise.

The order given by the variable ordering process to the set of unscheduled request legs under consideration is the one which will dictate the order in which scheduling attempts for the requests will be made by the insertion heuristic. Our insertion algorithm views this ordered set of request legs as a sequence and determines subsequences of it which may be feasibly assigned to each resource.

Once a request leg order has been determined for scheduling, the insertion process can begin. If it is determined that a particular request leg may be supported by a given mission, the point of insertion into the existing schedule remains to be determined.

**Column Generation**

At the local scheduler level, mission schedules are represented as event lists. Each even corresponds to a take-off or landing of the aircraft, either for service or to support a travel request. To each event there corresponds time window constraints (ET, LT). These bounds can be viewed as an extension of the constraint hierarchy, propagated down from the time constraints on the mission and up from the time windows on the requests being serviced by the event. The ET and LT values for an event are created and maintained in a manner similar to the forward pass and backward pass for maintaining ET and LT values in the Critical Path Method (CPM) [4]. Each insertion of a travel request onto a mission results in possibly creating new events, adding the constraints imposed by the new request to those already in place, and recalculating the ET and LT for events in the event list, using the forward and backward method of CPM to propagate constraints across the bottom level of the constraint hierarchy. A time window becomes violated by an insertion when it produces an ET value for an event which is greater than its corresponding LT value. Only those insertions which do not violate any of the existing time windows for events are feasible. If an insertion of a request is deemed feasible, value ordering is performed on the values associated with all feasible insertions of the request leg into the resource schedule.

The generation of feasible task-to-resource assignments can be viewed as a constraint satisfaction problem. Assignments which are feasible are exactly those which can be performed without violating the constraints on the task, on previously scheduled tasks, or on the resource. From the sequence of unscheduled request legs, a SPP column and associated cost are generated for each subsequence which may be feasibly assigned to a specific resource. As assignments of tasks to resources are considered, the effects of time windows for the new tasks propagate to those of existing flight legs. This alteration

of flexibility of aircraft mission, within the allowable bounds of the resource. The effects of committing to a particular change in a mission schedule will propagate to the time windows of other missions for the same aircraft and ultimately to the schedule for the entire fleet. Likewise, alterations in the scheduling constraints at the fleet level affect the time windows for take-offs and landings to accommodate a particular request leg. Thus the fleet schedule structure may be viewed as a dynamic model, in which alterations at any level propagate throughout the entire system, affecting constraints on each component. Only those alterations that preserve the satisfaction of all component. Only those alterations that preserve the satisfaction of all previously established constraints are considered feasible. Figure 1 illustrates how the insertion of a request leg on a mission can ultimately affect the entire fleet schedule.

The upper levels of the hierarchy in figure 1 represent information which must ultimately be available globally in a distributed paradigm. We assume, in our insertion heuristic, that the schedule and allowable time window for a specific mission are a local concern at the point when scheduling takes place. This allows the scheduling agent the autonomy needed to manipulate the schedule locally within constraints imposed at the time the mission is selected for scheduling. It further allows us to develop an insertion heuristic for local scheduling which may be extended implemented within a distributed scheduling system.

## DISTRIBUTED COMPUTING MODELS FOR ARAS

A model for decentralized airlift resource allocation and scheduling (DARAS) must provide basic information support needed for the individual schedulers to perform their tasks. Moreover, integrity of data must be maintained as reservations are created and edited by multiple schedulers. The model must provide a means of performing distributed scheduling which is both efficient for the agent and effective in producing globally desirable schedule structures. Three basic models that provide varied facilities for information sharing are discussed. These are referred to as the "Autonomous peer process", "Cooperative peer process", and "Client-server" models. These three models are listed in order of increasing suitability for implementing a distributed decision support system for multi-agent scheduling. We discuss them in general and then make recommendations for DARAS, in particular.

### Autonomous peer processes

In this model all schedulers execute identical processes which allow them to enter new requests for travel, create/edit reservations for aircraft and generate reports by directly accessing the underlying database, as if each scheduler was the exclusive scheduling agent. Acquisition of available aircraft resources is permitted on a first come, first served basis, and individual schedulers have no information provided by the software support environment concerning resource needs of other schedulers. The advantage to this model is that algorithmic extensions from the single agent case are simple. This model is sometimes employed when multiple schedulers are stationed in close proximity to each other, and use verbal agreements to prevent much contention from arising. A disadvantage is a need for a sophisticated database management system to maintain data integrity during concurrent use of the system by multiple schedulers. Heavy contention for resources will cause inefficient scheduling and may cause scheduling rollbacks, due to lacking or incomplete knowledge of concurrent scheduling efforts. The lack of a global view limits possibilities for providing meaningful algorithmic decision support in the scheduling process.

In order for individual scheduling processes to be informed of potential contention for a given aircraft, communication between all processes must be established and information about resources of mutual interest must be shared between the appropriate processes. Lacking this communication, efforts to minimize schedule rollback will require a locking mechanism to reserve aircraft, thereby precluding other agents from accessing them. Since the scheduling process may take considerable time, locking is not scalable and the model was rejected as an alternative for the current work. Requiring that each scheduling agent communicate scheduling intentions with every other scheduler places an undue burden on both the schedulers and the communication network. It is clear that providing mechanisms for sharing of resource requirement information between schedulers requires a model in which relevant data is easier to collect, maintain, and disseminate.

### Cooperative peer processes

This model was proposed by Sycara, et. al. [7] for the distributed job shop scheduling problem. In that environment, shop "stations" or machines, each able to service a single requested task at a time, are scheduled to complete "orders" composed of requirements for processing by one or more stations. The basis of the model is the establishment of a mechanism which allows processes to communication resource needs and scheduling intentions with other processes that share a need for particular resource.

In terms of airlift scheduling, where a resource may service several requests simultaneously, we make each scheduler process a "node" in a distributed scheduling system. Each scheduler agent becomes the resource manager of a set of aircraft and is the only agent enabled to make changes to the schedules of those aircraft it

controls. An agent considering employing a particular aircraft, in a schedule being constructed locally, must request current information on the availability of that resource from its manager, and in so doing must "register" an interest in that aircraft. The resource manager then updates the aggregate demand information for that particular aircraft and provides it to all registered scheduling agents. Finally, any scheduler requiring a resource must send a request to the resource manager who, up receipt of a reservation request, verifies that the aircraft is available, makes appropriate database accesses to update schedules, and then shares the updated information with the "interested" schedulers. This model can clearly be implemented on a single multi-processing machine, but can also be extended to a network of computers.

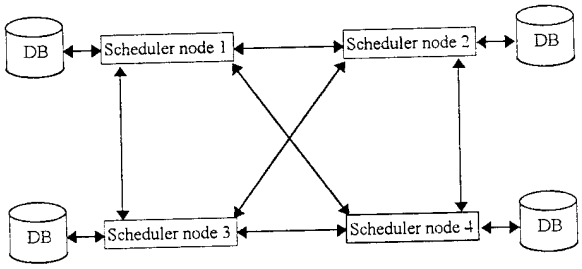


Figure 2: An example of the connections required for a fully connected four-node cooperative peer process network.

As an example, Figure 2 shows the communication connections for a fully connected network of four scheduler nodes. The number of connections maintained is $O(n^2)$, where n is the number of participating nodes. Communication can be minimized by only communicating information from a resource manager to those nodes which have indicated an interest in a particular aircraft. The model also makes fewer demands on the database management system, for concurrency control, by requiring that only an agent responsible for a resource can update its schedule and related information. The underlying database can be either centralized or distributed. Support for local decision making with attention to global resource needs becomes achievable. However the implementation of global optimization strategies are difficult because there is no view which encompasses all aircraft schedules. The loss of communication with one or more of the nodes in the network makes the resources maintained by the node(s) inaccessible to the rest of the schedulers. This is not likely in the single machine, multiple process case, but becomes an issue in the decentralized case. The problem can be mitigated by having backup nodes which take over the "down" nodes responsibilities, which further complicates the model.

**Client-server model**

In this model, the server is a process that waits to be contacted by a client process, and then services the client's request. In the context, of the executive airlift application, the client processes are the scheduler application processes, and the server acts as a resource manager. The server maintains allocation and contention information about aircraft, seats and times each craft is available, providing it to those scheduling agents requesting a particular resource. It controls access, maintains aggregate demand information, and communicates with "interested" nodes for all aircraft.

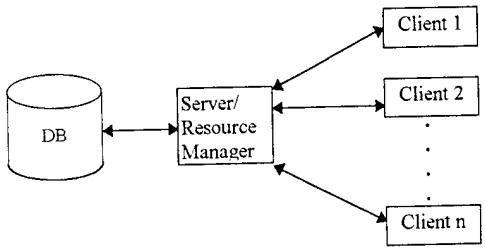


Figure 3: The client server model with n (scheduler) clients.

As with the cooperative peer processes model, scheduling agents are required to register intentions with the server before being granted access to or information about a particular aircraft and its schedule. Reservation requests for updating an aircraft schedule are sent to the resource manager, which has sole access to the database and can provide update information to those schedulers concerned with that particular resource. As figure 3 indicates, the number of communication connections in this topology is O(n) for a model with n clients. The server process can provide either iterative or concurrent service to their clients. Each type has advantages and disadvantages, depending on the application.

*The Iterative Server*

The iterative server queues messages from clients and processes them in a FIFO manner. This model works well when it is known that processing a message takes a small, relatively constant amount of time. If the number of simultaneous accesses to the server increases with the number of clients, this model will not scale well. Long waits in queue or refused messages will degrade performance at client nodes. However, there are advantages to this model, if the client access load is relatively low, as is the case when clients obtain resource information from the server access to the data base requires a more simple form of concurrency control in the DBMS. The iterative server handles all requests for resource availability and contention and is therefore in an ideal position to maintain a current view of resource allocation. This type of server is well suited for the implementation of global optimization strategies in distributed systems.

*The Concurrent Server*

A concurrent server, upon receipt of a client request, begins a new (child) process, dedicated exclusively to the service of the request. Upon completion of its tasks, the child process terminates. The parent server process is therefore available to accept requests from clients while other requests are being serviced. In this way it is possible to service multiple client requests. In the case of a large number of client processes attempting simultaneous access to the server, this model will, on the average, begin the service to a client in a more timely fashion. Database accesses will, however, occur from one of multiple server clones and will require greater concurrency control than the iterative server model. Furthermore, unless a mechanism for sharing up-to-date information between the server clones is implemented, management of a global view of resource availability becomes more difficult, and hence it is difficult to provide global optimization support. This can be mitigated by the implementation of shared memory access and a global optimizer process accessed by all the server clones.

**The Client Server Paradigm for DRAS**

Of the three models investigated, the client-server model provides for the most information being distributed to appropriate nodes with the least amount of communication overhead. In the airlift scheduling environment, where we expect a relatively small number of agents working simultaneously, the iterative server is the most appropriate, due to the natural way in which global optimization strategies can be implemented. Inherently a network model, the client-server model can be naturally implemented on either a single machine or a network of machines. The communication protocol for this model on a single machine is simple because communications failures between processes are not an issue. When extending to a network, the protocol must take into account failure at a node and network failure, though node failure is not as damaging as in the cooperative peer process model. If the server is expected to make all database accesses, the protocol must be designed to accommodate a wide range of server responses. These can include complete aircraft reservation schedule structures, for all aircraft, for a given schedule period. A significant challenge to implementing this model is providing the server and clients with strong decision support and optimization strategies.

**Implementation of the Client Server Model for DRAS**

A fundamental motivation for choosing the client server model is the desire to provide a global view to the scheduling process and to control access to commonly utilized resources. Database accesses by clients, either for information about resources under consideration or for database update, are therefore a natural point at which to introduce the separation of client and server responsibilities. The client process continues to function at the local level as it does in the single agent paradigm, however all database activities are now directed to the server process. By empowering the server with ultimate control of database accesses, we insure both database integrity and the ability of the server to maintain a global view of the demand for resources.

In designing the single agent model, we anticipated the extension to a networked environment and consequently, we implemented a design which would seamlessly incorporate a network interface. The clients' database access function calls are intercepted by a network interface which redirects them to the server process. Figures 4 a) and b) demonstrate the insertion of the network interface and server process into the direct database access for the autonomous scheduler.

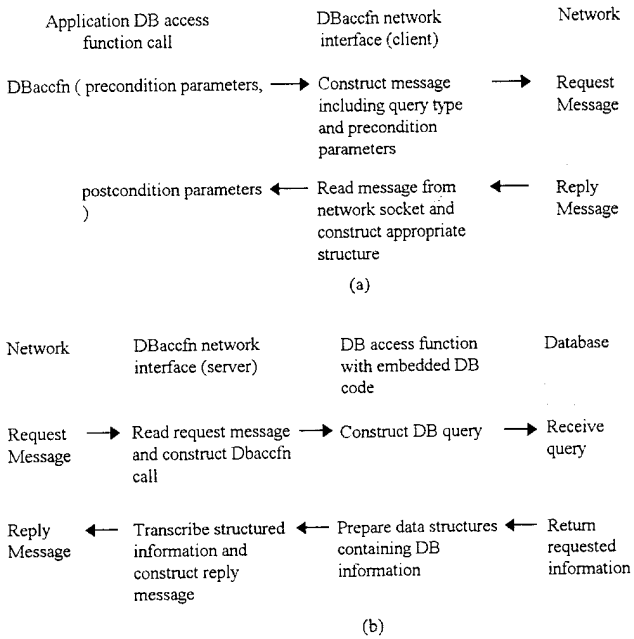


Figure 4: The unobstrusive extension of the local single scheduler application to the distributed paradigm involves inserting client and server network interfaces between the application's DB access function call and the DB access function.

Once the network interface has been established, it is possible for the single scheduling agent client to make the same database requests and updates that were made directly. Now, however, the server is responsible for intercepting those requests and making accesses to the database. This puts it in a position to coordinate allocation of shared resources and propagate updated information regarding resource availability and constraints to its clients. The level of coordination provided by the server may vary from simple advisory information on resource availability and contention to autocratic determination of specific resource assignments. At minimum, a reservation system, or